

CLAIMS

What is claimed is:

- 1 1. A computer system, comprising:
 - 2 a pipelined, simultaneous and redundantly threaded ("SRT") processor having at least two
 - 3 threads;
 - 4 an input/output ("I/O") controller coupled to said processor;
 - 5 an I/O device coupled to said I/O controller; and
 - 6 a system memory coupled to said processor;
 - 7 wherein said SRT processor further comprises:
 - 8 a load/store execution unit having a store queue, the store queue adapted to store
 - 9 memory requests submitted by the at least two threads, where said memory requests change values
 - 10 in system memory directly or indirectly;
 - 11 a compare logic coupled to said load/store execution unit;
 - 12 wherein said compare logic scans the contents of said store queue for corresponding
 - 13 memory requests, and said compare logic verifies that each corresponding memory request
 - 14 matches; and
 - 15 wherein said compare logic, based on whether the corresponding memory requests
 - 16 match, performs one of 1) allowing the memory request to execute, and 2) initiating fault recovery.
- 1 2. The computer system as defined in claim 1 wherein said memory requests that directly or
- 2 indirectly change data values in system memory further comprise at least committed store requests.

1 3. The computer system as defined in claim 1 further comprising said SRT processor capable
2 of performing, within each thread independently, program instructions in an order different from
3 the other thread.

1 4. The computer system as defined in claim 1 wherein each of said threads of said processor
2 performs speculative branch execution independently from the other.

1 5. A method of checking for transient faults in a simultaneous and redundantly threaded
2 processor having at least two threads, the method comprising verifying, as between the at least two
3 threads, only committed store requests and data load requests from sources that are not cached.

1 6. The method as defined in claim 5 further comprising:
2 storing a first committed store from a first of the at least two threads;
3 storing a second committed store from a second of the at least two threads;
4 comparing at least an address and data field of the first written committed store to at least
5 an address and data field of the second store; and
6 allowing at least one of the committed stores to execute if the address and data of each of
7 the first and second stores exactly match.

1 7. The method as defined in claim 6 further comprising:
2 disallowing execution of either of the first or second committed stores if their address and
3 data fields do not exactly match; and
4 initiating a fault recovery sequence.

1 8. A method of detecting transient faults in a simultaneously and redundantly threaded
2 microprocessor having at least two threads, the method comprising:
3 executing a program as a first thread;
4 generating a first committed store request from said first thread;
5 storing said first committed store request in a storage queue;
6 executing the program as a second thread;
7 generating a second committed store request from said second thread;
8 storing said second committed store in said storage queue;
9 checking an address and data associated with said first committed store request against an
10 address and data associated with said second committed store request in a compare logic; and
11 allowing one of said first and second committed store requests to execute if the checking
12 step shows those committed store requests are exactly the same.

1 9. The method as defined in claim 8 wherein executing the first and second threads further
2 comprises executing the first thread a plurality of program steps ahead of the second thread.

1 10. The method as defined in claim 9 further comprising:
2 allowing the first and second threads to make speculative branch execution independent of
3 each other.

1 11. The method as defined in claim 9 further comprising:
2 allowing the first thread to execute program steps out of an order of the program;

3 allowing the second thread to execute program steps out of the order of the program; and
4 allowing each of the first and second threads to execute the program in a different order from each
5 other.

1 12. A simultaneous and redundantly threaded microprocessor comprising:
2 a first pipeline executing a first program thread;
3 a second pipeline executing a second program thread;
4 a store queue coupled to each of said first and second pipelines;
5 a compare circuit coupled to said store queue;
6 wherein each of said first and second program threads independently generate
7 corresponding committed write requests, and each thread places those committed write requests in
8 the store queue; and
9 wherein said compare circuit detects transient faults in operation of said first and second
10 pipeline by comparing at least the committed store requests from each thread.

1 13. A pipelined, simultaneous and redundantly threaded ("SRT") processor, comprising:
2 a fetch unit that fetches instructions from a plurality of threads of instructions;
3 an instruction cache coupled to said fetch unit and storing instructions to be decoded and
4 executed; and
5 decode logic coupled to said instruction cache to decode the type of instructions stored in
6 said instruction cache;
7 wherein said processor processes a set of instructions in a leading thread and also in a
8 trailing thread, and wherein the instructions in the trailing thread are substantially identical to the

T 06 10 55 28 00

1 instructions in the leading thread, the instructions in the trailing thread beginning processing
2 through the processor after the corresponding instructions in the leading thread begin processing
3 through the processor;

4 and wherein said processor detects transient faults by verifying as between the leading and
5 trailing threads only the committed stores and uncached memory read requests.

1 14. A method of detecting transient faults in a simultaneous and redundantly threaded
2 microprocessor having at least two threads, the method comprising:

3 executing a program as a first thread;

4 generating a first committed store request from said first thread;

5 storing said first committed store request in a storage queue;

6 executing the program as a second thread;

7 generating a second committed store request from said second thread;

8 checking an address and data associated with said first committed store request against an
9 address and data associated with said second committed store request; and

10 allowing one of said first and second committed store requests to execute if the checking
11 step shows those committed store requests are exactly the same.

1 15. The method as defined in claim 14 wherein executing the first and second threads further
2 comprises executing the first thread a plurality of program steps ahead of the second thread.

1 16. The method as defined in claim 15 further comprising:

2 allowing the first and second threads to make speculative branch execution independent of
3 each other.

1 17. The method as defined in claim 15 further comprising:

2 allowing the first thread to execute program steps out of an order of the program;
3 allowing the second thread to execute program steps out of the order of the program; and
4 allowing each of the first and second threads to execute the program in a different order from each
5 other.

1 18. A simultaneous and redundantly threaded microprocessor comprising:

2 a first pipeline executing a first program thread;
3 a second pipeline executing a second program thread;
4 a store queue coupled to at least said first pipelines;

5 wherein each of said first and second program threads independently generate
6 corresponding committed write requests, at least said first thread places the committed write
7 requests in the store queue; and

8 wherein said second thread detects transient faults in operation of said first and second
9 pipeline by comparing at least the committed store requests from each thread.